REMARKS

Entry of this amendment along with reconsideration and allowance of the subject application are respectfully requested.

Claim 16 stands rejected under 35 U.S.C. §112, second paragraph. In particular, the Examiner suggests amendment to clarify that the interpreter does not execute native program instruction words. Claim 16 has been amended to specify that the processing unit executes both the native program instruction words and the interpreted program instruction words, which should obviate the Examiner's concern. Withdrawal of the rejection under 35 U.S.C. §112, second paragraph is respectfully requested.

Claims 1-5, 8 and 19 stand rejected under 35 U.S.C. §102(e) as being anticipated by U.S. Patent 6,513,156 to Bak. This rejection is respectfully traversed.

A description of Bak's system, Bak's objectives, and distinctions of the independent claims from Bak's teachings are set forth in the prior response. Notwithstanding those distinctions, the Examiner maintains the anticipation rejection. In an effort to address concerns raised by the Examiner in paragraph 3D of the Office Action, claims 1 and 19 have been amended to specify that the return address used as a pointer to the interpreted code portion is generated *during execution* of the native code call instruction.

In maintaining the anticipation rejection based on Bak, the Examiner relies on Figure 11 (and, in particular, step 907) and the text at column 12, lines 13-17 as allegedly teaching the features of claim integers (v), (vi), and (vii) of claims 1 and 19. This reliance is misplaced.

Bak discloses in column 11, line 42 to column 12, lines 22, the following sequence of operations.

1) A current bytecode pointer is saved.

2) An interpreter return address is pushed onto the stack memory.

3) Snippet code for the invoke_virtual function is executed. The snippet comprises a compiled native machine instruction equivalent to "call<function>", in which the address of the desired virtual function is hard-coded in the native machine instruction (see column 11, lines 50-65).

4) One the virtual function finishes execution, the system returns to the return address that was pushed onto the stack.

5) Native instructions are executed to reload the save bytecode pointer.

6) The system continues to execute non-native code from the point where the interpreter left-off.

Bak does **not** disclose that the "return address is generated during execution of said native code call instruction." Instead of generating a return address during a single native code call instruction execution, Bak discloses that a "snippet" performs the sequences 1)-6) listed above. Since the "snippet code" comprises a plurality of instructions, one of the stages of Bak's Figure 11 can be the counterpart of the feature of claim 1 whereby a native code call instruction is executed. Because the "native code call instruction" of claim 1 involves a single instruction, rather than multiple instructions, the counterpart of this feature in the system of Bak must be stage 905 of Figure 11 where a jump to the virtual function is performed.

The Examiner contends that the counterpart in Bak of the return address of claim 1 is the return address that was pushed onto the stack at step 907 of Figure 11. This is an instruction executed subsequently to the native function call instruction corresponding to step 905 of Figure 11. Indeed Bak, column 12, lines 13-15 specifies that only once the virtual instruction *finishes execution* does the system return to the return address that was pushed on the stack at step 907

(counterpart of the return address of claim 1). Thus, it is clear that Bak teaches away from the claimed feature where the return address is generated *during execution* of the native code call instruction.

Although Bak teaches that the return address 907 is generated during execution of snippet code, as illustrated by Figure 11, it is clear that that snippet code represents not a single native code call instruction, but a series comprising a plurality of instructions. Indeed, the Examiner concedes that this is the case by specifying in the Advisory Action that "the snippet will not only hold the native machine instructions...." That concession is further evidenced by column 12, lines 15 and 16, which specify that the interpreter performs the step of reloading the saved bytecode pointer. This precludes the possibility that the multiple stages of Figure 11 correspond to a single native machine instruction, since an interpreter would not be required to execute a native instruction at stage 905 so that at least stages 905 and 909 are distinct instructions. Also, stage 805 of Figure 10 of Bak indicates that the snippet code is formed from a plurality of instructions.

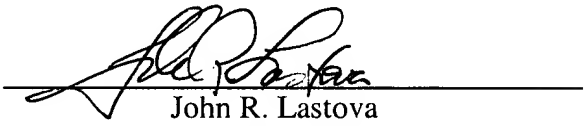
The present invention has the advantage that rather than requiring several native program instruction words to carry out the task of switching from executing native program instruction words to executing interpreted program instruction words, a single special purpose native program instruction word in the form of a native function call performs the switch. The return address of the native code call instruction is used to pass the location of the interpreted program instruction words to the interpreter. Bak's system suffers from the same disadvantages as the prior art, acknowledged on page 2, lines 27 and 28 of the specification, since the snippet code of Bak that is used to replace the invoke virtual bytecode comprises a plurality of program instructions which are required to perform the switch between native code and interpreted code.

Thus, although Bak fails to disclose elements (v)-(vii) of the independent claims as explained in the after final response, there is no question that Bak fails to disclose or suggest element (vi) of claims 1 and 19. Accordingly, the anticipation rejection should be withdrawn. Moreover, the obviousness rejections set forth in numbered paragraphs 10 and 12 rely on secondary references which do not remedy the deficiencies of Bak with respect to the independent claims.

This application is now in condition for allowance. An early notice to that effect is earnestly solicited.

Respectfully submitted,

NIXON & VANDERHYE P.C.

By: 
John R. Lastova
Reg. No. 33,149

JRL:at
1100 North Glebe Road, 8th Floor
Arlington, VA 22201-4714
Telephone: (703) 816-4000
Facsimile: (703) 816-4100